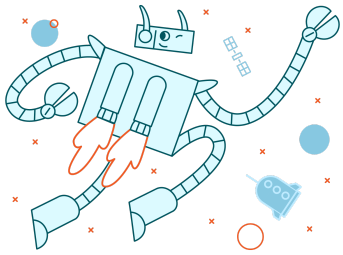
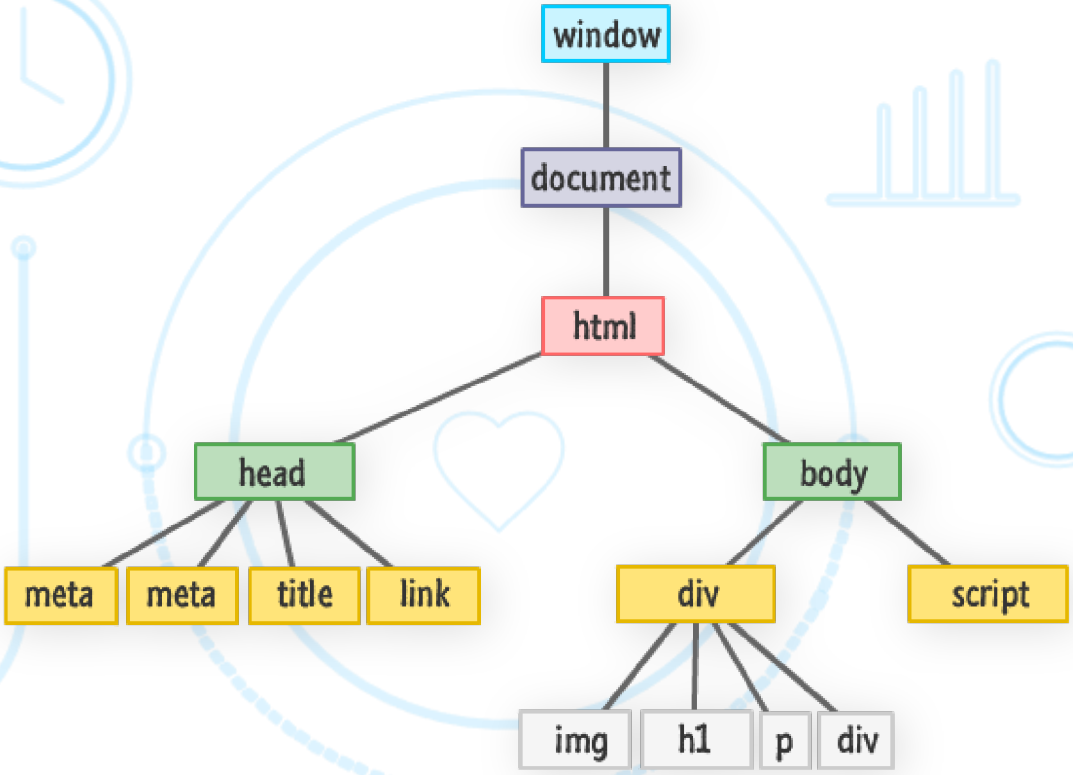
Advanced DOM



When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of objects:



Each HTML element in the image is called a **node**. And **nodes** can have **children**. For example, `div` and `script` are **children** of the `body` **node**.

Note that only data nodes directly below and connected to a node are **children**. Other nodes are out of the **children** scope.

Get children nodes

JavaScript has the keyword `children` to get the children of an element.

```
// Get body children
document.body.children
// Get an id-specific element children
document.getElementById("id-here").children
```

Iterate over children

Remember using `for...of` loops to iterate over arrays? You can use it to iterate over children nodes too as they are an array-like structure.

```
let childrenText = []

for (const element of document.body.children) {
    // Print the HTML tags alone
    console.log(element.tagName)

    // Print children elements' innerHTML
    console.log(element.innerHTML)

    // Print children elements' text contents
    console.log(element.innerText)

    // Push children element's inner texts into the
array
    childrenText.push(element.innerText)
}
```

Select children

JavaScript provides some methods to select children.

`firstElementChild` returns the first element in the array.

`lastElementChild` returns the last element in the array.

`children[number]` returns the element based on its position.

Webpage:

```
<ul id="shopping-list">
    <li>Pasta</li>
    <li>Tomatoes</li>
    <li>Basil</li>
</ul>
```

```
let listEl = document.getElementById("shopping-
list") // returns an array of elements
listEl.firstElementChild // <li>Pasta</li>
listEl.lastElementChild // <li>Basil</li>
listEl.children[1] // <li>Tomato</li>
```

Select nodes by tag names

We can use `getElementsByTagName("tag-name")` to select a group of similar html tags. The result will always be an **array-like** data type.

Note that tag names are **case insensitive**.

```
let imageElements =
document.getElementsByTagName("img")

for (const imgElement of imageElements) {
    console.log(imgElement.src) // Prints out the
URL in each image's src attribute
}
```

ID vs Class

Recall each `id` must be unique in a web document. However, mutiple HTML elements can be given the same `class` to group them together.

```
<p class="special">Special</p>
<p class="special">Special</p>
<p class="normal special">Extra special</p>
<p class="normal">Normal</p>
```

Get elements by class

Use the selector `getElementsByClassName("class-name")` to get elements by their class.

```
let classElements =
document.getElementsByClassName("special") // Gets
all elements with class = "special"

let multiClassElements =
document.getElementsByClassName("normal special") //
Get all elements with class = "normal" AND class =
"special"

console.log(classElements[0].innerText) // Prints
the text content of the first element
console.log(multiClassElements[0].innerText)
```

Chaining selectors

Selectors like `getElementById`, `getElementsByTagName` and `getElementsByClassName` can be chained when needed to select a specific element.

Web document:

```
<div id="container">
    <p class="special">Hello</p>
    <p>world</p>
</div>
<p class="special"></p>
```

Selecting only the `<p class="special">` inside `<div>`

```
let classEl =
document.getElementById("container").getElementsByCl
assName("special")
```

Further reading - Creating elements dynamically

You can create html elements inside JavaScript and add them to the webpage by attaching them as a child node.

`document.createElement("tag-name")` creates an element of the given tag.

`.innerHTML` add text to the element (will overwrite previous text).

`.appendChild` add the newly created element and its content into the HTML document as a **child** node.

```
<div id="header"></div>
<p> my head is missing, AHHHHH</p>
```

```
// Create new element
let title = document.createElement("h1")

// Assign content to the new element through its
attributes
title.innerHTML = "My head is in JavaScript, AHHHH"
title.style.backgroundColor = "red"

// Append new element as a child node of the element
with id="header"
document.getElementById("header").appendChild(title)
```